Answer to Discussion/observation of Problem 6

1.
```
Number of keys = 100000

HashMap average insert time = 7.60
ArrayList average insert time = 0.80
LinkedList average insert time = 3.10

HashMap average search time = 1.80
ArrayList average search time = 6537.80
LinkedList average search time = 26088.60
```

2.
```
Number of keys = 100000

HashMap average insert time = 7.40
ArrayList average insert time = 0.80
LinkedList average insert time = 4.10

HashMap average search time = 1.20
ArrayList average search time = 7031.80
LinkedList average search time = 26062.20
```

3.
```
Number of keys = 100000

HashMap average insert time = 7.40
ArrayList average insert time = 0.80
LinkedList average insert time = 4.10

HashMap average search time = 1.20
ArrayList average search time = 7031.80
LinkedList average search time = 26062.20
```

Upon running the program several times, and calculating the estimate of the execution time for both insert and search over the three data structures HashMaps, ArrayLists, and LinkedLists. I found that the ArrayList with the faster insertion runtime compared to the rest two, then LinkedList, and then HashMap, but all three had relatively similar insertion runtimes. However, when it comes to the search runtime, HashMaps are outperformed with a significant faster runtime compared to ArrayLists and more to LinkedLists. HashMap is very efficient at lookup like with the containsKey() method checking if a key exists or retrieving a value based on a key, those operations we will expect a time complexity of O(1) on average if no collision occurs or we handles the collision well, which signifinicant faster than ArrayList and LinkedList which we search uses the contains() method as it is an implementation of indexOf(), so they will iterate through the whole list in a worst scenario that runs O(n).